PUC-Rio – Software Básico – INF1018
Prova 2 – 25/06/2015 – Turma 3WB

Atenção: Não esqueça de colocar o seu nome e número de matrícula na(s) folha(s) de respostas.
Boa prova!

1. (1,5 pontos) Considere o módulo `calcula.c` abaixo:

```
#include <math.h>
#include <stdio.h>
double calculo_magico(double um, double dois);

extern double valores[];
extern int num_valores;

int flag_fim = 0;
static int prox = 0;

double calcula_um (double um_valor) {
  double v,s;
  if (prox == num_valores) {
    printf("terminei o calculo de %d valores\n", num_valores);
    flag_fim = 1;
    return INFINITY;
  }
  s = sin(um_valor);
  v = valores[prox++];
  return calculo_magico(s, v);
}
```

   Liste quais símbolos do módulo objeto `calcula.o` apareceriam como **D** (símbolo na área de dados, exportado), **T** (símbolo na área de código, exportado) e **U** (símbolo indefinido, referência externa) na saída do programa `nm`.

2. (3,0 pontos) Traduza a função `bar` abaixo para assembly IA-32 do gcc/Linux (visto em sala), utilizando as regras básicas de passagem de parâmetros, retorno de resultado e uso de registradores em C.

   (Não se preocupe se você não entender o que a função faz, apenas traduza-a literalmente).

```
void   foo(double *v, int n);
double boo(double x);

double bar(float ini) {
  double local[2];
  double *pl = local;
  double sum = (double) ini;
  int i;

  foo(pl, 2);
  for (i = 0; i < 2; i++, pl++)
    sum += boo(*pl);
  return sum;
}
```

3. (2,5 pontos) Considere o trecho de código C

```
int fun1(double d, float f, int i);
int fun2(...) {
  int i;
  i = fun1(X,Y,Z);
  ...
}
```

onde **X**, **Y**, e **Z** são valores constantes de tipos `double`, `float` e `int`, respectivamente.

Suponha que no momento em que o controle chegar ao label `fun1` (ou seja, imediatamente após a CPU executar `call fun1` e antes de começar a executar o código da função `fun1`), o valor de `%esp` seja `0xffffa2b0` e que, inspecionando o conteúdo da pilha de execução a partir deste endereço, encontramos os seguintes valores (exibidos abaixo em hexadecimal):

```
0xffffa2b0 - e0
0xffffa2b1 - 83
0xffffa2b2 - 04
0xffffa2b3 - 08
0xffffa2b4 - 00
0xffffa2b5 - 00
0xffffa2b6 - 00
0xffffa2b7 - 00
0xffffa2b8 - 00
0xffffa2b9 - 00
0xffffa2ba - f8
0xffffa2bb - bf
0xffffa2bc - 00
0xffffa2bd - 00
0xffffa2be - 5a
0xffffa2bf - 41
0xffffa2c0 - fc
0xffffa2c1 - ef
0xffffa2c2 - ff
0xffffa2c3 - ff
```

Quais são (em notação decimal) os valores das constantes **X**, **Y**, e **Z**?

(ATENÇÃO: mostre como você chegou aos valores exibidos. Valores sem contas **NÃO valem ponto!**)

4. (2,0 pontos) Traduza para assembly IA-32 (convenções gcc/Linux) a função `abre` a seguir. **Atenção**: Sua função *assembly* não deve chamar as funções `rename` e `open` da biblioteca. Ao invés disso, ela deve **chamar diretamente o Sistema Operacional!**.

```
#include <stdio.h>
#include <fcntl.h>
int exists(char *arq);

int abre(char *old, char *new) {
  if (exists(old))
    rename(old, new);
  return open(new, 0);
}
```

Para construir as chamadas ao sistema, lembre-se que o código da chamada ao SO é passado em `%eax` e os demais parâmetros em `%ebx`, `%ecx`, `%edx`, etc... A interrupção que aciona o sistema operacional é a `0x80`.

O código da chamada `open` é **5** e o código da chamada `rename` é **38**.